The Internet of Things (Part 4)

IoT Security

System and data security should be of utmost importance to any professional engineer who designs devices for the Internet of Things (IoT). This month Bob looks at security issues related to the IoT and what embedded systems designers can do to secure their devices.

By Bob Japenga

This morning, my wife and I were talking with a friend and his wife about the Internet of Things (IoT). I know. I know. Aren't there better things to talk about on a beautiful day with dear friends? But I am a techie, and since you are reading this article, so are you. My friend said that he couldn't see any reason to have his refrigerator or his toilet connected to the Internet. When I, with my impeccable techie logic, persuasively convinced him of the reason why refrigerators and toilets will connect to the Internet, he responded by saying that he would not allow such devices into his home because of security issues. Touché! The way things are right now in the infancy of the IoT, he is correct. We, the designers of the IoT devices have a lot to learn about security. Over the next couple of months, we will look at some of the ways the security of existing IoT devices has been compromised so that we can get better at this.

In July of 2015, many of us shuddered when we read about the Jeep Cherokee that could be controlled remotely by a hacker. Admittedly these weren't every day hackers: they were working on a grant to attempt to demonstrate security holes in the automotive industry. These two men, Charlie Miller and Chris Valasek, hacked into a Jeep with the driver's permission to demonstrate a series of massive security vulnerabilities. They could control the A/C, set the volume and station selection of the radio, turn on the windshield wipers and even put their picture on the screen. All of this without ever touching the Jeep. They then remotely downloaded a hacked version of the firmware into the car, which enabled them to do much more serious things: turn off the brakes, stop the transmission from functioning, and jiggle the steering wheel. All of this was done without the owner of the car granting them any physical access. Shuddering yet?

Thankfully, the car manufacturer (Fiat-Chrysler) quickly responded to close the most egregious holes. But as we look at what they did wrong, we can improve the security of the IoT devices that we design. As we do this we have to keep in mind the warning that Miller and Valasek gave at the beginning of their talk in August 2015: "Don't ever say that your system is hack proof." With enough time and money, I agree with them that all of our systems could be hacked. Our job is to make it extremely expensive in time and money to accomplish the fact. That is where we are going over the next few articles. How can we make the IoT devices we design be more secure?

TOP SECURITY VIOLATIONS

Think of security as the protection of a medieval castle. You have the moat. You have the door and the wall if they get by the moat. And you have the boiling oil if they get through the door. Finally, you have your valiant warriors willing to fight to the death to protect the castle. To improve the security of your IoT device, you need many layers of protection. This doesn't mean that there is only one entrance to the castle. We will talk about underground tunnels and the like next time. Although the Miller and Valasek presentation is full of many interesting details, the major areas where Fiat-Chrysler went wrong are quite simple: an open port that allowed hackers to telnet into the box (the moat); no authentication method once there was the open port (the wall or door); a firmware update process that was not encrypted or signed (the boiling oil); and the critical data bus was not sufficiently protected (the fight to the death). Let's look at each one of these and see how we can make our IoT systems more secure.

OPEN PORTS

Miller and Valasek found that the Jeep left open port 6667 over both Wi-Fi and over the cell network. To their credit, in July 2015, the open port on the cell network was closed within days of the article exposing the vulnerability. Without this port, this means of remote access over the cell network was eliminated. The vulnerability over Wi-Fi still existed (needs a recall to correct), but the hacker would need to be within 30 m of the car to have remote access. A hacker would need to trail the targeted car for several hours while attempting to hack the WPA password and know the approximate year of manufacturer. This doesn't pose a significant threat to the entire fleet.

So what does it mean to leave a port open? Let's first make sure we understand what a port is. Devices on the Internet communicate with each other over a protocol called IP. Under this protocol each device has an IP address. For every address, the device can listen and talk on many ports. When you use your browser to point at a website with HTTP, it connects to the web server listening on port 80. For secure sites using HTTPS, it uses port 443. If you telnet into a device you usually connect to port 23. Each daemon may listen on one or more ports. To leave a port open means that a process on your device is listening on that port. In the case of the Jeep, a telnet process in the car was listening on port 6667 and allowed telnet access over that port without any authentication.

Most likely this port was left open by mistake. This is easy to do when we use complex operating systems like Linux or QNX (as was used on the Jeep). The OS may leave open ports that you the designer did not intend to leave open. So what's the moral of the story? One, always check to see if there are any ports left open unintentionally. Two, if you most leave a port open, make sure that access requires authentication.

AUTHENTICATION & PASSWORDS

This leads to the issue of authentication and passwords. There has to be a very good reason to design a system that allows the user named: root (Linux and Unix systems) to login to a device in the field. It not only gives the hacker a well-known username, but also provides more privileges than you probably want to give to someone remotely. So your first line of defense, once the moat is crossed, is the username and password (your wall and door).

Passwords, if used properly can provide a significant deterrent to hackers and a moderate level of security. In the case of the Jeep, a secret user name and a strong password would have shut down this attack completely.

Here is my take on passwords for embedded devices exposed to the outside world. This is your next line of defense. Your spikes on the wall. Most of us designers don't access our embedded devices very often when they are deployed. When we do access them, we have tools that can allow us to use large and random type passwords. The next question is: Should it be a secret password or unique to each machine or both? The advantage of a secret password is that it can be random. However, one based on the MAC or serial number is algorithmic and can be "guessed." There are advantages and disadvantages of each based on the size of your organization and other factors which are beyond the scope of this article.

Once created, how big should the password be? In my opinion, eight cryptographically random characters of all of the printable ASCII characters is more than enough protection. You can ignore those stories of hackers making 400 billion guesses per second. Your telnet (shudder) or SSH login should be configured to take several seconds for a machine to try one guess. So if your login has that, you can use one guess per second in your calculation.

ABOUT THE AUTHOR

Bob Japenga has been designing embedded systems since 1973. In 1988, along with his best friend, he started MicroTools, which specializes in creating a variety of real-time embedded systems. With a combined embedded systems experience base of more than 200 years, they love to tackle impossible problems together. Bob has been awarded 11 patents in many areas of embedded systems and motion control. You can reach him at rjapenga@microtoolsinc.com.

If you create an eight cryptographically random character password, there are 6.09 quadrillion possible passwords to try. And 6.09 quadrillion seconds is a long time (more than 193 million years).

The moral of the story for Jeep was to use user secret authentication and use strong passwords. Even with an open port, authentication and a password would have stopped this attack.

REMOTE UPDATES

One of the beauties of the IoT is that remote software updates cover a multitude of our sins. With over-the-air updates we can update every part of our code safely and reliably. But what about securely? Miller and Valasek showed that the Jeep design had a number of major flaws in its software update procedure. As a result, they were able to update one of the processors in the Jeep to allow them to modify devices on the CAN bus. The Jeep's line of defense was shattered. The moat was crossed; the wall breached and the door was opened to enemy troops. Once they could do this, they could do almost anything with the car. So what can we learn from the update process?

First, the update was neither encrypted nor signed. With public key encryption, the hackers would need your secret private key to encrypt the new program they wish to send to your device. Without the correct encryption, the software would reject the update. With signing, the device will only accept new software if it is from a trusted source with some sort of digital signature. At our company we do both with our software updates.

Once this is done, the software should allow an update of the public key or the digital signature. Authentication can escape or leak out of a company in a flash. You don't want a security breach (releasing of your private keys or passwords) to cause all of your security to come crashing down like a house of cards.

DATA BUS PROTECTION

Now that the moat was crossed; the wall breached; the door opened, there was one last thing that Fiat-Chrysler could have done to save the castle. With their own program running in the Jeep, Miller and Valasek's were able to create a new packet on the CAN bus to control the Jeep. However, the data packet was checked by a very unique algorithm that they couldn't figure out. Fiat-Chrysler appeared to plan for such a security breach because the checksum of the data packet on the bus was built with a non-trivial secret algorithm. Someone was thinking about the possibility of a hack. However, because the code was not protected, they could reverse engineer the code, find the algorithm thus enabling them to create new packets on the CAN bus. Moral of the story: encrypt your code. Because they can pull the chip and extract the code, pick a chip that allows you to lock the code so that it cannot be read if the chip is removed.

SERIAL LAYERS

In days of old, when kings were trying to protect their castles they provided several layers of protection. If you got over the moat, there was the wall. If you got over the wall and opened the door there was the hot oil poured on your head. To improve security on our IoT designs we need to have this kind of serial layers of protection. If they breach the network and find an open port, they have to break the authentication. If they breach the authentication, they do not have a means to change executables. You get the idea. Next time we will look at another security debacle and see what we can learn from it in order to make our IoT devices more secure. Of course, only in thin slices.

circuitcellar.com/ccmaterials

RESOURCES

DEFCONConference, "DEF CON 23—Charlie Miller & Chris Valasek—Remote Exploitation of an Unaltered Passenger Vehicle," 2015, www.youtube.com/watch?v=OobLb1McxnI.

A. Greenberg, "Hackers Remotely Kill a Jeep on the Highway—With Me in It," 2015, www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/.

C. Miller and C. Valasek, "Remote Exploitation of an Unaltered Passenger Vehicle," 2015, http://illmatics.com/Remote%20Car%20Hacking.pdf.

Wired.com, "Hackers Remotely Kill a Jeep on the Highway—With Me in It," www.youtube.com/watch?v=MK0SrxBC1xs.